FYS-STK4155 - PROJECT 1:
REGRESSION ANALYSIS AND RESAMPLING METHODS

ALEXANDER HAROLD SEXTON
ERIK JOHANNES HUSOM

OCTOBER 7, 2019

Abstract. In this project we apply the Ordinary Least Squares method, Ridge regression and Lasso regression, in combination with the Bootstrap resampling technique, on two different data sets: Data generated from the Franke function and a topographical terrain data set. We compared the three different regression methods, and the OLS method gave us the lowest means squared error on test data in both cases, with respectively $\mathrm{MSE}_{\mathrm{Franke}} = 0.0136$ and $\mathrm{MSE}_{\mathrm{terrain}} = 0.0091$.

Contents

1. Introduction

Regression methods are a set of commonly used tools in machine learning, and the overarching goal is to study the dependence of a target or response variable on a number of independent variables, often called predictors. In order to perform regression we need a data set that contains information about the phenomenon we want to study, and this is used to estimate a function f, which is assumed to represent the relationship between the predictors and the response. The result is a model which can be used to understand how the response variable is related to the input variables, or just simply to predict the outcome based on new input data.

The aim of this project is to analyze different regression methods, specifically the Ordinary Least Squares (OLS) method, Ridge regression and Lasso regression. The OLS is one the most common approaches when performing linear regression, and produces a set of polynomial coefficients that describe the model as a function. Ridge and Lasso regression are two types of shrinkage methods that, in contrast to OLS, may reduce or "shrink" some of the coefficients in the model, which may be desirable if some predictors are more important than others, or if some of the predictors in a data set are correlated.

Another important aspect of this project is to explore resampling of data. We have specifically looked at the resampling technique called *bootstrap*, where we draw random samples from a training data set multiple times, and refit a model with each of the samples. By doing this we can extract additional information about the data set.

In the first part of this project, we used a generated data set based on a function called the Franke function, and analyzed the performance of our chosen regression methods, including resampling of the data. In the second and last part, we applied the same techniques on topographical terrain data from an area of Norway.

This report includes the theory used in the project, along with a short review of the methods we implemented in our source code. Then we present our results, followed by a discussion, and lastly a conclusion where we sum up the most important findings.

2. Theory

2.1. Ordinary Least Squares.

Suppose we have a collection of data consisting of a set of observations $\mathbf{y} = [y_0, y_1, .., y_{n-1}]$, which are assumed to be the response of a set of variables $\mathbf{x} = [x_0, x_1, .., x_{n-1}]$, described by some unknown function f. With no underlying knowledge of f, and a motivation to predict values of y which are not in the original data set, it is usual to assume that there is a linear relationship between $\mathbf{x}$ and $\mathbf{y}$. A parameterization of f can be chosen based upon some insight of the behaviour of $\mathbf{y}$ wrt. $\mathbf{x}$, and in the simplest case a polynomial of degree p, that is

$$y = y(x) \to y(x_i) = \tilde{y}_i + \epsilon_i = \sum_{j=0}^{p} \beta_j x_i^j + \epsilon_i \tag{1}$$

where β_j is an unknown parameter and ϵ_i is the error in our prediction y_i, which accounts for influences on our prediction values from other sources than our predictors x_i. This gives rise to the set of equations

$$\begin{aligned}
y_0 &= \beta_0 + \beta_1 x_0^1 + \beta_2 x_0^2 + \cdots + \beta_{n-1} x_0^p + \epsilon_0 \\
y_1 &= \beta_0 + \beta_1 x_1^1 + \beta_2 x_1^2 + \cdots + \beta_{n-1} x_1^p + \epsilon_1 \\
y_2 &= \beta_0 + \beta_1 x_2^1 + \beta_2 x_2^2 + \cdots + \beta_{n-1} x_2^p + \epsilon_2 \\
&\vdots \\
y_{n-1} &= \beta_0 + \beta_1 x_{n-1}^1 + \beta_2 x_{n-1}^2 + \cdots + \beta_{n-1} x_p^{n-1} + \epsilon_{n-1}
\end{aligned}$$

We can rewrite this as a linear algebra problem by defining the vectors

$$\begin{aligned}
\mathbf{y} &= [y_0, y_1, y_2, ..., y_{n-1}]^T \\
\boldsymbol{\beta} &= [\beta_0, \beta_1, \beta_2, \ldots, \beta_p]^T \\
\boldsymbol{\epsilon} &= [\epsilon_0, \epsilon_1, \epsilon_2, \ldots, \epsilon_{n-1}]^T
\end{aligned}$$

and the matrix

$$\boldsymbol{X} = \begin{bmatrix} 1 & x_0^1 & x_0^2 & \ldots & x_0^p \\ 1 & x_1^1 & x_1^2 & \ldots & x_1^p \\ 1 & x_2^1 & x_2^2 & \ldots & x_2^p \\ \vdots & \vdots & \vdots & \ldots & \vdots \\ 1 & x_{n-1}^1 & x_{n-1}^2 & \ldots & x_{n-1}^p \end{bmatrix} \tag{2}$$

where $\mathbf{X} \in \mathbb{R}^{n \times p}$ is known as the *design matrix*[1]. Equation 1 can then be written as

$$\boldsymbol{y} = \tilde{\boldsymbol{y}} + \boldsymbol{\epsilon} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \tag{3}$$

Ordinary Least Squares is a method of estimating the parameters $\boldsymbol{\beta}$, which minimizes the distance between our target y_i and the prediction $\tilde{y}_i$. A measure of this fit is the mean squared error [1, p. 29]

$$\text{MSE}(y_i, \tilde{y}_i) = \frac{1}{n} \sum_{i=0}^{n-1} (y_i - \tilde{y}_i)^2 \tag{4}$$

which we define as our *cost function*

$$C(\boldsymbol{\beta}) = \frac{1}{n} (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^T (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}) \tag{5}$$

[1]In general, we are not restricted to a polynomial of only one variable. For a polynomial of degree p with two variables, the design matrix would consist of every permutation of $x^i y^{p-i}$ along each row.

Since this is a convex function of $\boldsymbol{\beta}$, it has a unique minimum which occurs when its gradient is equal to zero. By differentiating wrt. $\boldsymbol{\beta}$ and setting it equal to zero, we obtain

$$\frac{\partial C(\boldsymbol{\beta})}{\partial \beta} = -\frac{2}{n}\boldsymbol{X}^T(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta}) = 0 \tag{6}$$

$$\boldsymbol{X}^T\boldsymbol{y} = \boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta}, \tag{7}$$

assuming that $\boldsymbol{X}^T\boldsymbol{X}$ is positive definite. This gives us the optimal values for the parameters $\boldsymbol{\beta}$:

$$\boldsymbol{\beta} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} \tag{8}$$

The predicted outputs $\tilde{\boldsymbol{y}}$ for a given input $\boldsymbol{X}$ is then given by

$$\tilde{\boldsymbol{y}} = \boldsymbol{X}\boldsymbol{\beta} = \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T \tag{9}$$

The predicted values $\tilde{\boldsymbol{y}}$ can be viewed as and orthogonal projection of $\boldsymbol{y}$ onto the vector space spanned by the columns of $\boldsymbol{X}$. The variance of the coefficients β is

$$\begin{aligned}
\mathrm{Var}(\beta) &= \mathrm{Var}\big((\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}\big) &(10)\\
&= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\mathrm{Var}(\boldsymbol{y})\big((\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\big)^T &(11)\\
&= \mathrm{Var}(\boldsymbol{y})(\boldsymbol{X}^T\boldsymbol{X})^{-1} &(12)
\end{aligned}$$

2.2. **Ridge and Lasso Regression.** There are several ways we can try to improve the Ordinary Least Squares method. The coefficients produced by the OLS method may often have a low bias and a large variance (to be discussed in the following subsection), and it treats all parameters in the same way without taking into account that some predictors may be more important than others. We can improve the prediction accuracy by removing or reducing some of the coefficients, and Ridge regression is a method that shrinks the coefficients according to a penalty on their size. We introduce a *tuning parameter* $\lambda \geq 0$, which quantifies the amount of shrinkage. For Ridge regression, our cost function is[2, p. 64]:

$$C(\boldsymbol{\beta}) = (\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta})^T(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta}) + \lambda\boldsymbol{\beta}^T\boldsymbol{\beta} \tag{13}$$

By taking the derivatives wrt. $\boldsymbol{\beta}$, as we did with the Ordniary Least Squares method, we obtain that the optimal values for the parameters $\boldsymbol{\beta}$ are:

$$\boldsymbol{\beta} = (\boldsymbol{X}^T\boldsymbol{X}+\lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{y}, \tag{14}$$

and the predicted outputs $\tilde{\boldsymbol{y}}$ for a given $\boldsymbol{X}$ is

$$\tilde{\boldsymbol{y}} = \boldsymbol{X}\boldsymbol{\beta} = \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X}+\lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T \tag{15}$$

We can see that setting $\lambda = 0$ results in the Ordinary Least Squares method.

Another shrinkage method is the Lasso (Least Absolute Shrinkage and Selection Operator) method, which uses the cost function[1, p. 219]

$$C(\boldsymbol{\beta}) = (\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta})^T(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta}) + \lambda\sqrt{\boldsymbol{\beta}^T\boldsymbol{\beta}} \tag{16}$$

The difference between Ridge and Lasso regression is that we replace the L_2 ridge penalty $||\boldsymbol{\beta}||_2^2 = \boldsymbol{\beta}^T\boldsymbol{\beta}$ with the L_1 Lasso penalty $||\boldsymbol{\beta}||_1 = \sqrt{\boldsymbol{\beta}^T\boldsymbol{\beta}}$. This altered penalty means that the solutions in the y_i becomes nonlinear, and we cannot find a closed form expression for the parameters. Instead, the Lasso method uses gradient descent to find the minimum of the function[2]. When using the L_1 norm instead of the L_2 norm we may end up with some coefficients being set to zero.

The tuning parameter λ needs to be selected carefully, and can be tuned by using for example cross-validation or Bootstrapping.

2.3. **Error analysis.** In order to evaluate the fit of the different models in this project, we use several indicators. One of the most common error measures is the *mean squared error* (MSE), which is already defined in equation 4. The MSE will be small if the predicted values are close to the true values, which is the ideal scenario.

Another measure is the R^2 score[1, p. 70]:

$$R^2(y,\tilde{y}) = 1 - \frac{\sum_{i=0}^{n-1}(y_i-\tilde{y}_i)^2}{\sum_{i=0}^{n-1}(y_i-\bar{y})^2}, \tag{17}$$

where $\bar{y}$ is

[2]The specific details of the theory behind the Lasso method and gradient descent algorithms is outside the scope of this project, which is also why we use the Python library Scikit Learn to perform this type of regression.

$$\bar{y} = \frac{1}{n}\sum_{i=0}^{n-1} y_i. \tag{18}$$

The R^2 score tells us how much of the variance in the response variable that is explained by the model. The value is ideally between 0 and 1, and a number close to 1 generally indicates that there is a strong relationship between the input variables and the response.

2.4. **Bias-Variance Tradeoff.** Bias and variance are two important quantities when it comes to predictive models. The bias is a measure of the difference between our prediction and the true value that we want to predict. Models with high bias tends to underfit the data, i. e. the model is oversimplified. Variance is an indicator of how sensitive the machine learning method is to the specific training data set. A high variance means that the model tends to be very affected by small fluctuations and noise in the predictors, and this leads in an overfitted model. As mentioned above, the quality of fit of a model is often measured using the MSE, which should be minimized, and we will now show how the MSE can be decomposed into three parts: Bias, variance and irreducible error.

Variance is in general defined as

$$\begin{align} \mathrm{Var}(x) &= \mathbb{E}\left[(x-\mathbb{E}[x])^2\right] \tag{19}\\ &= \mathbb{E}[x^2]-\mathbb{E}[x]^2 \tag{20}\end{align}$$

The bias of our prediction $\tilde{\boldsymbol{y}}$ can be mathematically written in the following way[2, p. 24]:

$$\begin{align} \mathrm{Bias}(\tilde{\boldsymbol{y}}) &= \mathbb{E}[\tilde{\boldsymbol{y}}-f] \tag{21}\\ &= \mathbb{E}[\tilde{\boldsymbol{y}}]-f \tag{22}\end{align}$$

The MSE, which is also the basis of the cost function for the OLS method, is given as:

$$\mathrm{MSE} = C(\boldsymbol{X},\boldsymbol{\beta}) = \frac{1}{n}\sum_{i=0}^{n-1}(y_i-\tilde{y}_i)^2 = \mathbb{E}\left[(\boldsymbol{y}-\tilde{\boldsymbol{y}})^2\right]. \tag{23}$$

We will assume that $\boldsymbol{y} = f(x)+\varepsilon$, where ε is the noise with zero mean and a variance of σ^2. This means that $\mathbb{E}[\varepsilon]=0$ and $\mathrm{Var}(\varepsilon)=\sigma^2$. We also have that $\mathbb{E}[f]=f$ since f is deterministic. Furthermore, the variance of $\boldsymbol{y}$ can be written as:

$$\begin{align} \mathrm{Var}(\boldsymbol{y}) &= \mathbb{E}\left[(\tilde{\boldsymbol{y}}-\mathbb{E}[\tilde{\boldsymbol{y}}])^2\right] \tag{24}\\ &= \mathbb{E}\big[(\boldsymbol{y}-f)^2\big] = \mathbb{E}\big[f+\varepsilon-f)^2\big] \tag{25}\\ &= \mathbb{E}\big[\varepsilon^2\big] \tag{26}\\ &= \mathrm{Var}[\varepsilon]+\mathbb{E}[\varepsilon]^2 \tag{27}\\ &= \sigma^2 \tag{28}\end{align}$$

In the last equality we use the rearranged version of equation 20: $\mathbb{E}[x^2]=\mathrm{Var}(x)+\mathbb{E}[x]^2$. The information above can then be used to decompose the MSE from equation 23:

$$\begin{align} \mathbb{E}\big[(\boldsymbol{y}-\tilde{\boldsymbol{y}})^2\big] &= \mathbb{E}\big[(f+\varepsilon-\tilde{\boldsymbol{y}})^2\big] \tag{29}\\ &= \mathbb{E}\big[(f+\varepsilon-\tilde{\boldsymbol{y}}+\mathbb{E}[\tilde{\boldsymbol{y}}]-\mathbb{E}[\tilde{\boldsymbol{y}}])^2\big] \tag{30}\\ &= \mathbb{E}\big[(f-\mathbb{E}[\tilde{\boldsymbol{y}}])^2\big]+\mathbb{E}[\varepsilon^2]+\mathbb{E}\big[(\mathbb{E}[\tilde{\boldsymbol{y}}]-\tilde{\boldsymbol{y}})^2\big] \tag{31}\\ &\quad + 2\mathbb{E}\big[(f-\mathbb{E}[\tilde{\boldsymbol{y}}])\varepsilon\big]+2\mathbb{E}\big[\varepsilon(\mathbb{E}[\tilde{\boldsymbol{y}}]-\tilde{\boldsymbol{y}})\big]+2\mathbb{E}\big[(\mathbb{E}[\tilde{\boldsymbol{y}}]-\tilde{\boldsymbol{y}})(f-\mathbb{E}[\tilde{\boldsymbol{y}}])\big] \tag{32}\\ &= (f-\mathbb{E}[\tilde{\boldsymbol{y}}])^2+\mathbb{E}[\varepsilon^2]+\mathbb{E}\big[(\mathbb{E}[\tilde{\boldsymbol{y}}]-\tilde{\boldsymbol{y}})^2\big]+2(f-\mathbb{E}[\tilde{\boldsymbol{y}}])\mathbb{E}[\varepsilon] \tag{33}\\ &\quad + 2\mathbb{E}[\varepsilon]\mathbb{E}\big[\mathbb{E}[\tilde{\boldsymbol{y}}]-\tilde{\boldsymbol{y}}\big]+2\mathbb{E}\big[\mathbb{E}[\tilde{\boldsymbol{y}}]-\tilde{\boldsymbol{y}}\big](f-\mathbb{E}[\tilde{\boldsymbol{y}}]) \tag{34}\\ &= (f-\mathbb{E}[\tilde{\boldsymbol{y}}])^2+\mathbb{E}[\varepsilon^2]+\mathbb{E}\big[(\mathbb{E}[\tilde{\boldsymbol{y}}]-\tilde{\boldsymbol{y}})^2\big] \tag{35}\\ &= (f-\mathbb{E}[\tilde{\boldsymbol{y}}])^2+\sigma^2+\mathrm{Var}\left(\tilde{\boldsymbol{y}}\right) \tag{36}\\ &= \mathrm{Bias}(\tilde{\boldsymbol{y}})^2+\sigma^2+\mathrm{Var}\left(\tilde{\boldsymbol{y}}\right) \tag{37}\end{align}$$

The decomposition above shows us that to minimize the total error, we need to have both low bias and low variance (σ^2 is the irreducible error), i. e. we want to have a model that captures the trends in the training data while also performing well on test data that have not been included in training. The problem is that when using methods that give low bias we usually end up with a high variance, and similarly, low-variance methods results in high bias. This is why we need to find the optimal balance between bias and variance when choosing what learning method to use.

2.5. **Resampling Methods.** In order to extract as much information from the data set as possible, we use resampling methods. The idea is to select subsets of the training data set repeatedly, and refitting a model each time. In this project we use a resampling method called the Bootstrap method. If we have a training data set $\boldsymbol{Z} = (z_1, z_2, \ldots, z_N)$ with $z_i = (x_i, y_i)$, we start by drawing a random dataset of size N (i.e. the same size as the original training set), but allow datapoints to be drawn several times (draw with replacement). Then we fit a model to this dataset with our chosen method, which in this project is different regression methods, and analyze the performance of the fit. We repeat this process B times, each time drawing a new data set from the original datset.

2.6. **Franke Function.** In order to analyze the different regression algorithms we used a sum of exponentials that is commonly used when testing fitting methods, called the Franke Function[3]:

$$f(x,y) = \frac{3}{4}\exp\left\{\left(-\frac{(9x-2)^2}{4} - \frac{(9y-2)^2}{4}\right)\right\} + \frac{3}{4}\exp\left\{\left(-\frac{(9x+1)^2}{49} - \frac{(9y+1)}{10}\right)\right\} + \frac{1}{2}\exp\left\{\left(-\frac{(9x-7)^2}{4} - \frac{(9y-3)^2}{4}\right)\right\} - \frac{1}{5}\exp\left\{\left(-(9x-4)^2 - (9y-7)^2\right)\right\}. \tag{38}$$

3. Methods

3.1. **Formatting the data.** We have two sources of data, one is generated by the Frankie function (equation 38), and the other is real terrain data in the form of elevation maps. In both cases, the prediction parameters are the x_1 and x_2 coordinates of the domain, and the height y is the output. For the Frankie data, we generate n uniformly distributed values of $x_1, x_2 \in [0,1]$ and place them in a mesh grid. We evaluate the function at each point in the grid, and add stochastic noise $\epsilon \in \mathcal{N}(0, \sigma^2)$ to each point.

In the case of the real terrain data[3], this is simply imported as a *.tif* file, giving the terrain height in meters, which is rescaled to be in units of kilometers. We also resize the image, such that only 1/16th of the original image is used. The original image is of dimensions 3600×1800 organized in a 2D array, where we have select the 900 first points in the x_1 and x_2 directions as our data set. Finally we slice every other point in this array, such that the final image is of dimension 450×450. This is done due to memory issues when operating on larger data sets.

We now initialize our design matrix by organizing the values of x_1 and x_2 according to equation 2, such that each column consist of a unique combination of powers $x_1^i x_2^{p-i}$, $i = 0, 1, \ldots, p$, and each row containing unique combinations of x_1 and x_2 values. We now have $\boldsymbol{X} \in \mathbb{R}^{n^2 \times p}$. The $\boldsymbol{y}$ values in our data are flattened to a 1D array, resulting in our target vector $\boldsymbol{y} \in \mathbb{R}^{n^2 \times 1}$.

Ideally one would want to normalize the data, which involves subtracting the mean of each column from the corresponding column in $\boldsymbol{X}$ and also dividing each column by its standard deviation. However, we have had varying results when trying to implement this, but have not viewed it as a big problem, as the values of the data generated by the Franke function is mostly in the region $y \in [0,1]$, which is also the case of the terrain data after converting from meters to kilometers.

3.2. **Regression Methods.** The main class in our implementation which performs the regression methods is called *RegressionMethods*, and is instantiated simply with a specification of the desired method, that is, either ordinary least squares, Ridge or Lasso. To train the model, the class has a method called *fit*, which takes as arguments the training data $\boldsymbol{X}_{train}$, and calls upon the relevant regression method. The ordinary least squares method finds the optimal $\boldsymbol{\beta}^{OLS}$ values by equation 8, and does this by using the Numpy library's matrix multiplication functionality and *linalg.pinv* function for finding the inverse matrix[4]. In the same manner, it computes the optimal $\boldsymbol{\beta}^{Ridge}$ by solving equation 14. In this case, the class must also be instantiated with the desired penalty parameter λ. This is also the case when using the class' Ridge regression method. We have not implemented this method ourselves, but instead uses the functionality of Scikit Learn's *Lasso* function. Any instance of the model can also predict a set of input parameters different from those it has been trained on, this is done by calling the *predict* method with an input parameter $\boldsymbol{X}_{test}$, which calculates the matrix-vector multiplication of $\boldsymbol{X}_{test}\boldsymbol{\beta} = \tilde{\boldsymbol{y}}$.

3.3. **Resampling.** Our resampling method of choice is Bootstrapping, which we have implemented such that it first splits our data into training and test data, then randomly selects points and fits the model to these selected points. When this process has been completed a desired number of times, it calculates the error metrics of the predictions on both the test and training data. Pseudo-code for the algorithm:

[3]The original data file can be found at the website `https://earthexplorer.usgs.gov` with Entity ID: SRTM1N41W001V3
[4]In cases where the inverse matrix does not exists, it instead computes the pseudo-inverse.

```
Split the data $\boldsymbol{X}$ and $\boldsymbol{y}$ into training data and test data
for ( $n_{bootstraps}$ ) {
    Select $n_{train}$ random samples from the training data
    Train the model on $\boldsymbol{X}_{train}$
    Test the model on $\boldsymbol{X}_{test}$
    Save all predicted values $\tilde{\boldsymbol{y}}$
}
Calculate mean MSE, $R^2$ score, bias$^2$ and variance
```

In order to find the optimal parameters which gives the best fit to our data, we have also implemented a function which performs the above algorithm for chosen model value ranges of λ and polynomial degrees d, and uses the MSE as a measure for how well the particular model performs.

3.4. **Benchmarks.** We have constructed a test case, which tests the results of our implementations of OLS and Ridge against the predictions of Scikit learn's methods. The data which is used in the test is generated from the Franke function, with 100×100 points. We then calculate the mean squared error between the prediction on this data produced by our code and Scikit learn's. In both cases the difference in the predictions was to machine presicion.

3.5. **Source code.** The source code of this project is written in Python, and can be found in the GitHub repository at `https://github.com/alexahs/FYS-STK4155/tree/master/Project_1`. The repository consists of the following files:

- RegressionMethods.py
- Resamplnig.py
- unitTests.py
- functions.py
- analysis.py
- main.py

4. Results

4.1. **Regression on generated data.** The performance of the ordinary least squares fit to the data is shown for different model complexities in figure 4.1 (left). The prediction on test data has a minimum mean squared error of MSE = 0.0136 and $R^2 = 0.819$ (not shown in the plot), which occurs when the model complexity is of degree $d = 5$, while the prediction on the training data has no minimum and keeps declining for higher complexity. The plot to the right in the same figure shows again the mean squared error of the predictions on the test data, and also its decomposition into bias^2 and variance. Figure 4.6 also shows the confidence intervals of the different β parameters for the optimal model with $d = 5$.

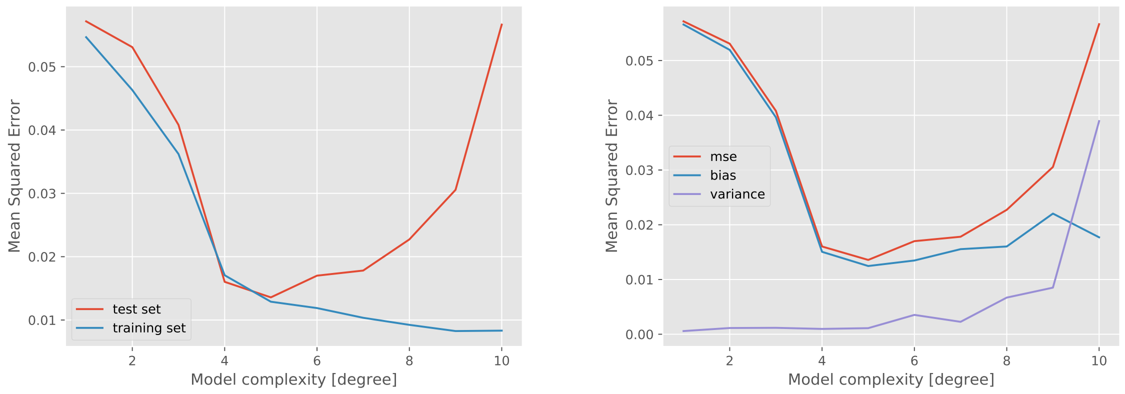

FIGURE 4.1. Error in the OLS regression prediction on the generated data as a function of the models polynomial degree for the training and testing data (left). Figure to the right shows the bias^2 and variance decomposition of the MSE in the prediction on the test data.

For Ridge regression, figure 4.2 shows the mean squared errors in the predictions on the test data for values of the hyper parameter λ and polynomial degree d. The minimum error is at $\lambda = 10^{-4}$ and $d = 9$, with MSE $= 0.0139$[5] and $R^2 = 0.874$. Figure 4.3 shows the performance of Ridge regression in a separate run for different polynomial orders d, with the optimal

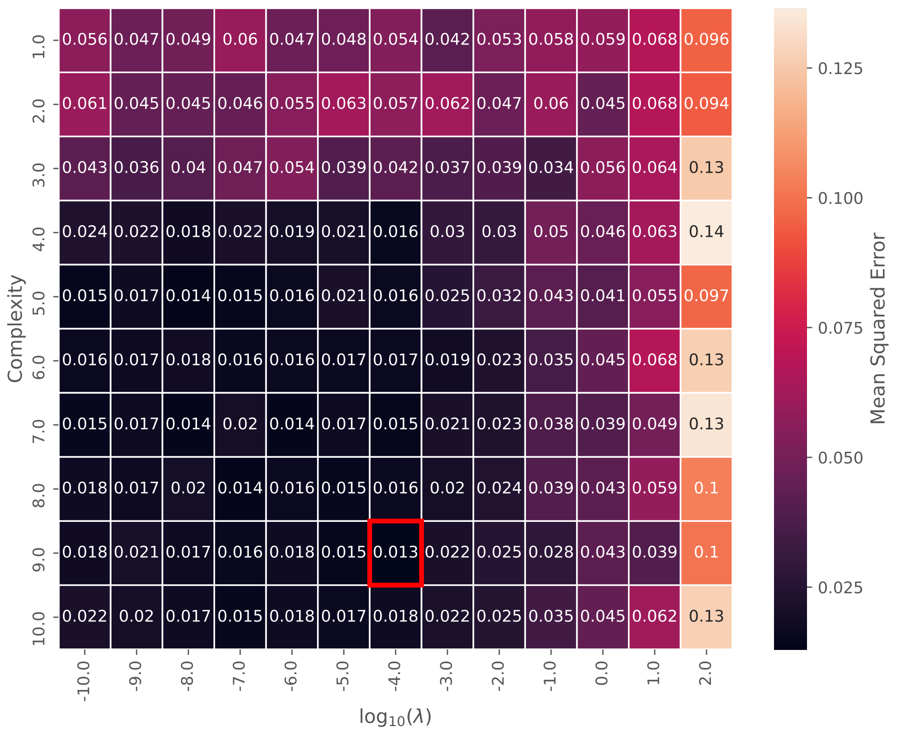

FIGURE 4.2. Heat map of the MSE of the predictions of Ridge regression on the generated data for different values of λ and model complexity d.

value of $\lambda = 10^{-4}$ found in the above figure (4.2). The figure shows the errors in both predictions on the test data and training data, as well as the bias^2 and variance decomposition of the error. Here we found the minimum predicted MSE to occur at $d = 6$, with MSE $= 0.151$ and $R^2 = 0.877$.

Figure 4.4 shows the analysis of Lasso regression applied to the Franke function, with respect to the model complexity and the hyper parameter λ. The minimum mean squared error, MSE $= 0.0183$ was found with a polynomial degree of $d = 5$ and $\lambda = 10^{-8}$, and had a corresponding R^2-score of 0.7874.

In figure 4.5 we see the Lasso regression performed for different polynomial degrees, for the optimal value $\lambda = 10^{-8}$. The plots shows the MSE for the training and test set, and how the MSE, bias^2 and variance behaves as a function of model complexity.

TABLE 4.1. Comparison of the models with the lowest MSE for each regression method, applied to the Franke function data set.

Method	degree	$\log_{10}\lambda$	MSE	R^2
OLS	5		0.0136	0.819
Ridge	6	-4	0.0139	0.877
Lasso	5	-8	0.0183	0.728

[5]In the plot we see that the MSE is truncated at 0.013, but an inspection of the data showed that the actual value was 0.0139.

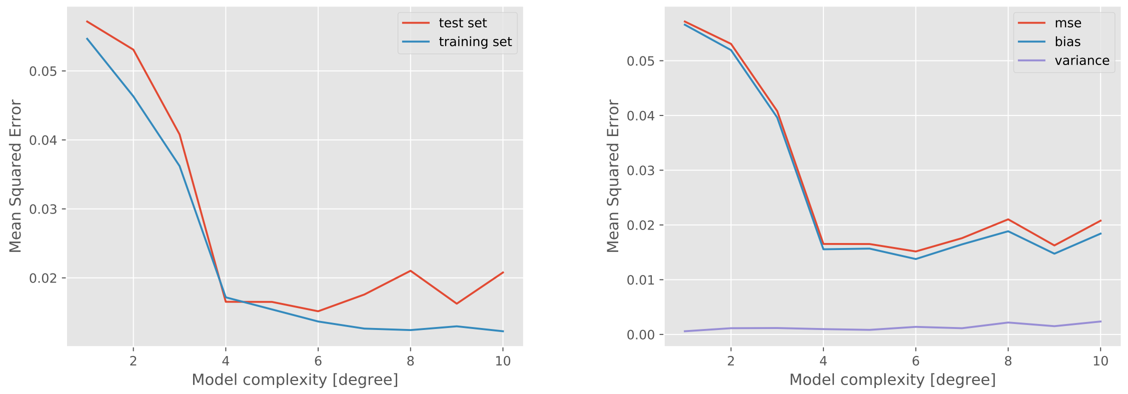

FIGURE 4.3. Error in the Ridge regression prediction on the generated data as a function of complexity, with $\lambda = 10^{-4}$. The left plot shows the error in the prediction on both the training and test data. The right plot shows the bias^2 and variance decomposition of the MSE on the test data.

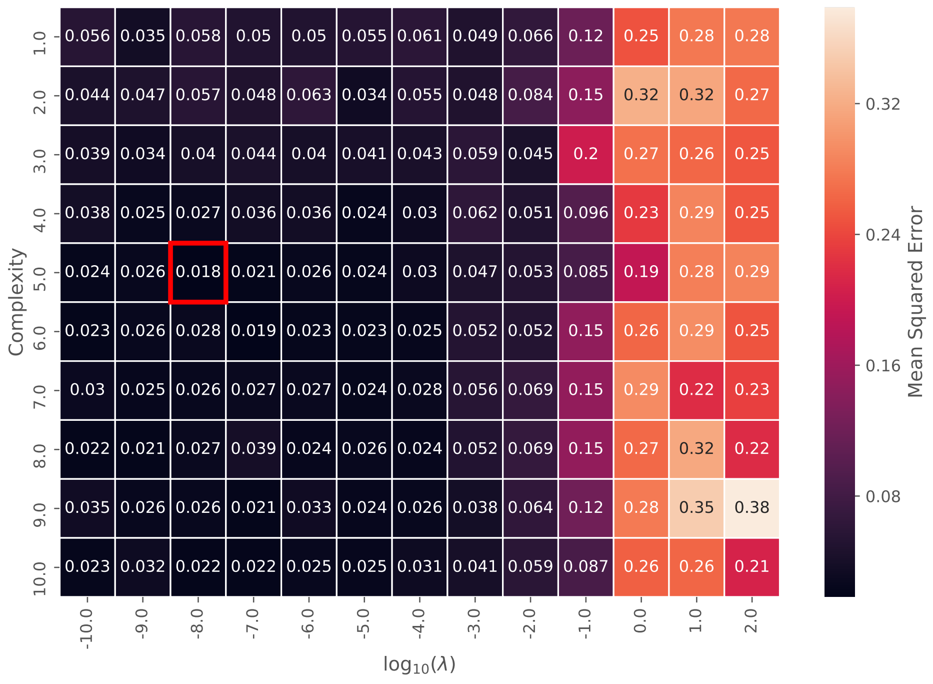

FIGURE 4.4. Heat map of the MSE of the predictions of Lasso regression on the generated data for different values of λ and model complexity d. Tolerance for convergence of the gradient descent: 0.0001, maximum iterations: 10000.

4.2. **Regression on real terrain data.** By doing an analysis of model complexities $d \in [1, 10]$ and hyper parameters $\log_{10}(\lambda) \in [-10, 2]$ with 100 bootstraps, we have in table 4.2 listed the predicted errors of the best model configuration of the three regression methods on the real data (see section 3.1). All three methods favor a polynomial of degree $d = 10$, with

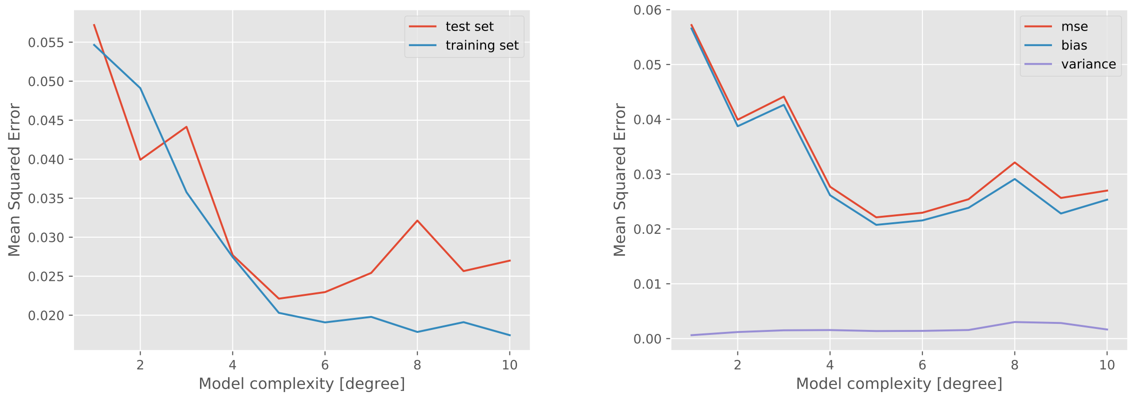

FIGURE 4.5. Error in the Lasso regression prediction as a function of complexity, with $\lambda = 10^{-8}$, when modeling on the Franke data set. The left plot shows the error in the prediction on both the training and test data. The right plot shows the bias^2 and variance decomposition of the MSE on the test data. Tolerance for convergence of the gradient descent: 0.0001, maximum iterations: 10000.

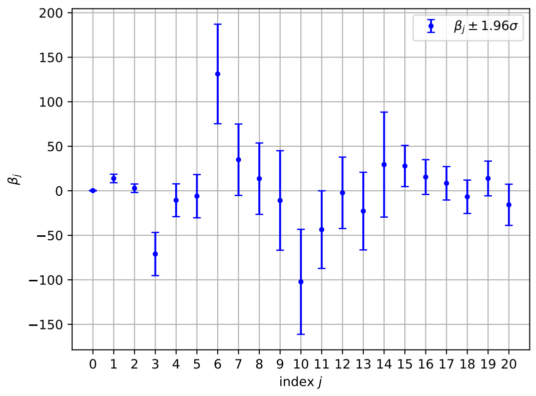

FIGURE 4.6. The parameters β_j for OLS with a confidence interval of 95%. Lower indices correspond to lower polynomial term coefficients, and vice verca.

OLS having the smallest estimate of the error, MSE $= 0.0091$. The Ridge and Lasso methods yield the best results with small values of the hyper parameter, $\lambda_{Ridge} = 10^{-9}$ and $\lambda_{Lasso} = 10^{-10}$ respectively. Figure 4.7 shows the bias-variance decomposition of the estimated errors for the three methods for different polynomial degrees. The curve representing the MSE (the red curve) lies directly underneath the curve of the bias^2, and is therefore not visible in the three plots. The confidence intervals of the OLS β parameters are shown in 4.8. In figure 4.9 we have plotted the data along side the predicted values of the OLS model with the optimal parameters found in table 4.2. The appendix (sec. 7) contains additional plots of the results of the tuning of the λ and d parameters on the terrain data for Ridge and Lasso, as well as the error in the predictions on the test and training data for all of the three methods.

5. DISCUSSION

5.1. **Regression on generated data.** The method of ordinary least squares gives the smallest estimation of the error, when predicting on the data generated by the Franke function. When studying the models behaviour for different complexities, we see from figure 4.1 (left) that the predictions on the test set has a minimum mean squared error MSE $= 0.0136$, which occurs at degree $d = 5$. The predictions on the training set keep declining for higher complexities, while the predictions on

TABLE 4.2. Comparison of the models with the lowest MSE for each regression method, applied to the terrain data set.

Method	degree	$\log_{10}(\lambda)$	MSE	R^2
OLS	10	-	0.0091	0.6323
Ridge	10	-10	0.0092	0.6325
Lasso	10	-9	0.0138	0.0157

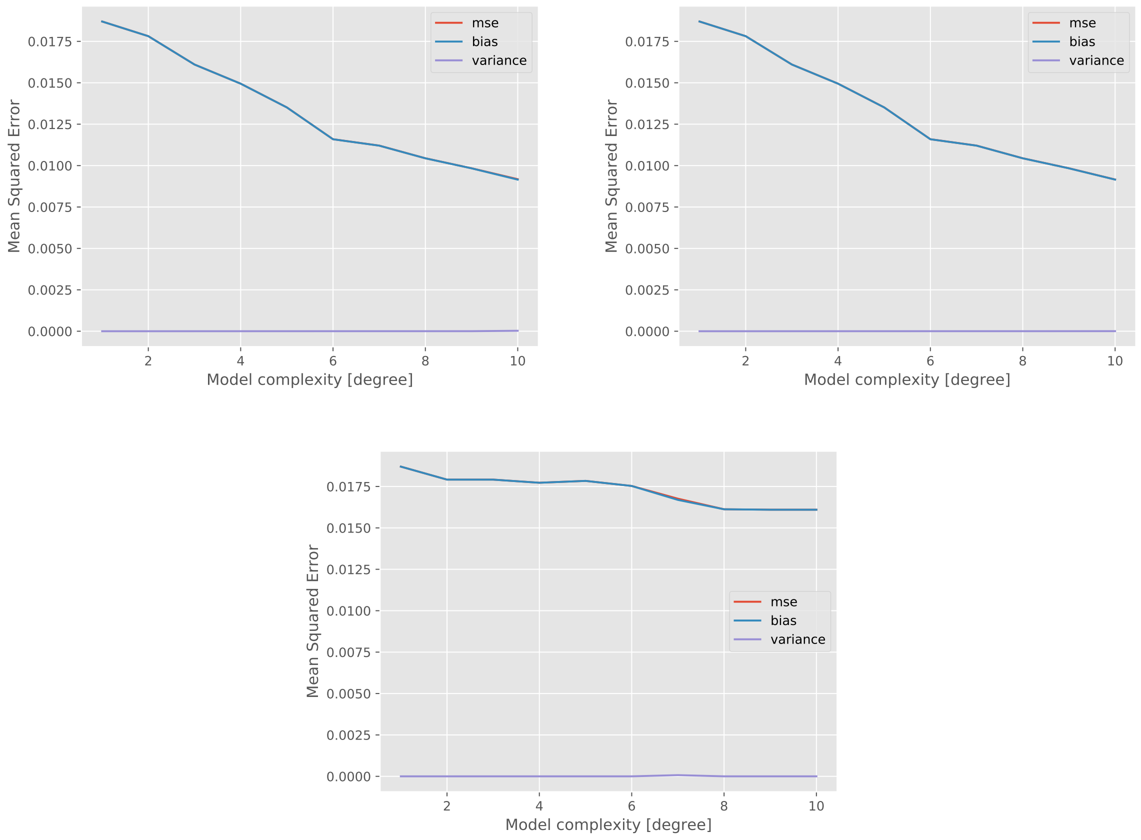

FIGURE 4.7. Estimated prediction errors of OLS (top left), Ridge (top right) and Lasso (bottom), and their bias^2 - variance decomposition as a function of model complexity.

the test set start to rise. When models of complexity $d > 5$ are introduced to data which it has not seen before, it fails to reproduce the testing data reliably. This suggests that the model has been fitted too closely to the training data and captured elements of noise, rather than the trends in the data itself.

Having a look at figure 4.1 (right) again, we see that in the area $1 \leq d < 5$ the error in the prediction consists solely of the bias term, while the variance remains almost non-existent. This is most likely due to the fact that the data cannot be captured accurately by a model with such a low complexity. The predictions by the model are so off target, relatively speaking, that the differences in the training and test data are mostly irrelevant compared to the model and the trends in the data. For $d > 5$, the variance term begins increasing, while the bias declines. This is most likely due to the model being, again, over fitted, and too sensitive to small fluctuations in the data set. Ideally we want to choose a model which is complex enough to capture the regularities in the training data, but also generalized well to data which it has not seen. As seen, this middle ground is struck with a polynomial of degree $d = 5$.

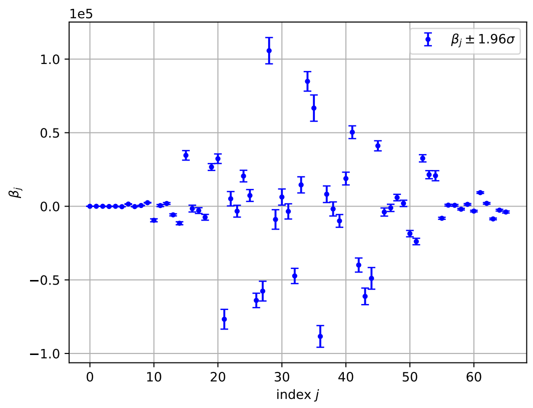

FIGURE 4.8. The parameters β_j for OLS with a confidence interval of 95%. Lower indices correspond to lower polynomial term coefficients, and vice verca.

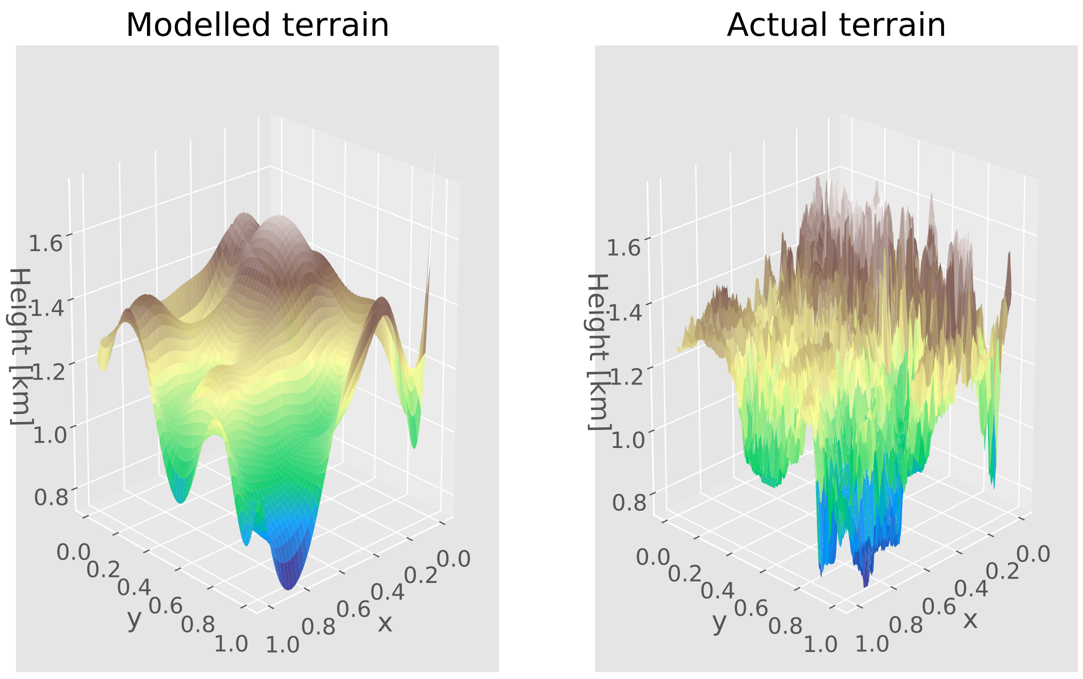

FIGURE 4.9. Height maps of the modelled terrain (left) and the terrain data(right).

From 4.1 we see that Ridge regression performs almost equally as well as OLS. With the optimal parameter $\log_{10}(\lambda) = -4$ and polynomial degree $d = 6$, the estimated error in the prediction is MSE = 0.0139. Having a look at the tuning of the parameters λ and d in figure 4.2, we see that its performance is generally good for small values of λ, which means that it is generally an OLS estimate, and may suggest that there is little correlation between the estimators in the model. When looking back at 4.3, where we did a separate analysis of the performance of Ridge with the optimal value for λ, we can see that we have the same problem with overfitting when the complexity of the model is higher than $d = 6$. In this plot, the minimum MSE occurs at $d = 6$, which is different from the analysis in 4.2, even though the models had the same parameters. A possible explanation for this is the random selection of data done by the Bootstrap resampling. We have chosen a relatively small number of 100 Bootstrap resamples, which may have resulted in a fluctuation favourable to one of the configurations. In retrospect, we could have looked at the histogram of the mean values of the randomly selected values, to ensure that it was normal.

When performing Lasso regression on the generated data, we see from the heat map 4.4 that the optimal model parameters which give the lowest prediction of the error MSE = 0.0183, are $\log_{10}(\lambda) = -8$ and complexity $d = 5$. We also see from the figure that for values of $\lambda < 1$, the model does a very bad job of reproducing the data. One reason for this may be that some of the coefficients β have been penalized too much to be able to reproduce the data in any meaningful way. This was also true for Ridge, but even more so for Lasso, which makes sense considering that the penalization parameter in Lasso regression is capable of shrinking the β parameters all the way to zero for greater values of λ. We have also had a closer look at the performance of the Lasso model in figure 4.5 with $\log_{10}(\lambda) = -8$, and plotted the MSE of the prediction for both the testing and training data. We see that the MSE of the test data has several local minima, which is quite interesting, and a global minimum at $d = 5$ with MSE = 0.0221. The trend of the training error keeps declining for Lasso as well, which is what we expect to see. The bias-variance decomposition of the test data error in the same figure (4.5) shows that the error is largely composed of the bias^2 term, and that the variance in the prediction is very low in comparison. In general, when comparing the Lasso method to OLS and Ridge, our results suggest that Lasso is a less suitable choice when trying to represent our data with a model.

5.2. **Regression on real terrain data.** Figure 4.7 shows the error analysis of the terrain data, with all three different types of regression methods. The OLS and Ridge regression gives very similar values for both the MSE, bias^2 and variance, which may indicate that the shrinkage imposed by the hyper parameter λ does not contribute much to the overall result when using Ridge regression. In fact, as we can see from table 4.2, the minimum MSE obtained with the OLS and Ridge method is 0.0091 and 0.0092 respectively. The OLS regression gives slightly better results, if we only use MSE as a measure of the quality of fit, but not by much. The third plot in figure 4.7 shows the error analysis for the Lasso methods, and it shows a similar trend of the three quantities (MSE, bias^2 and variance) as we have seen with the OLS and Ridge regression. The minimum MSE in this case is 0.0138 for a polynomial degree $d = 10$, and clearly performs worse than the OLS and Ridge regression. All three plots show a variance that is constantly very close to zero, which means that it contributes very little to the overall error, while bias, on the other hand, is shown to be the main source of error in the models.

Because of limitations in computational resources, we have only tested the models up to a polynomial degree of 10, but the error analysis in figure 4.7 suggests that for all methods, the error will decrease further for higher degrees, given the declining trend of the MSE. Since our models generally have a very low variance and a relatively high bias^2, it seems like the models are too simple to give a good representation of the terrain.

The confidence intervals of the parameters β_j, which is presented in figure 4.8, shows that the parameters with greater values generally have larger confidence intervals, which is what we would expect. The coefficients with index $j < 10$ have very small values. The β_j-values with the lowest and highest indices are very small compared to the rest, and indicate that the predictors they represent do not affect the model much. As mentioned above, the bias-variance plots for the terrain model suggest that fitting with an even higher polynomial degree may have resulted in a reduced error, but in the plot of the confidence intervals of β_j it seems like the highest order coefficients are relatively unimportant.

In figure 4.9 we see the real terrain data plotted next to our best-performing (in terms of having the lowest MSE) model, which was the OLS method with degree $d = 10$. A visual inspection tells us that the model seems to be following the main features of the terrain. The edges of the surfaces are easiest to inspect, and they match fairly well. Even so, the actual topographical data is very jagged, and our model gives an oversimplified and too smooth representation of the terrain.

In all of our analysis of the methods, the models have been fitted to the training data, but since the model configurations we have found to be best in each case is based on the predictions on the test data, the reported best model configurations is by extension fitted to the test data. It may therefore be wise to in stead consider what we have called the test data in this report as validation data, and have a separate set of test data which we finally predict our ideal model on, to give a less biased estimation of the models error.

6. Conclusion

The aim of this project was to investigate various regression methods and resampling techniques, by testing them on two different data sets, namely generated data from the Franke function and real topographical terrain data. We tested the Ordinary Least Squares method, Ridge regression and Lasso regression, all in combination with Bootstrap resampling. Among these we obtained the lowest test error with the OLS method, on both the Franke data set and the terrain data set. When performing regression on the Franke function, the minimum MSE occured with an OLS model of polynomial degree 5, which gave MSE = 0.0136. Our best terrain model had MSE = 0.0091, when using the OLS method with polynomial degree 10. This was also the most complex model we tested for the terrain data set, and our error analysis indicated that an even higher polynomial degree might have further reduced the mean squared error.

While we could expect that the Ridge and Lasso method would reduce the error compared to the OLS method, since they in theory should lower or remove the impact of less important predictors, our analysis show that in this project the OLS model gives the best performance on test data. One reason for this might be that we failed to do an optimal normalization of our data, which may have affected both the Ridge and the Lasso regression. Also, a more fine-tuned optimalization of the hyperparameter λ may have led to even better results. The minimum MSE-values of the Ridge regression were very close to those of OLS, and it is difficult to deem one method better than the other.

This project has shown how different types of regression methods perform on various problems, and how resampling techniques can be used to extract as much information as possible out of a data set. The error analysis we used, by observing both the mean squared error, the bias and the variance, has proven to be a very useful tool when selecting the optimal prediction model of a certain phenomenon.

References

[1] G. James, D. Witten, T. Hastie, and R. Tibshirani. *An Introduction to Statistical Learning*. Springer, New York, 2017.
[2] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, New York, 2009.
[3] R. Franke. *A Critical Comparison of Some Methods for Interpolation of Scattered Data*. Naval Postgraduate School, California, 1979.

7. Appendix

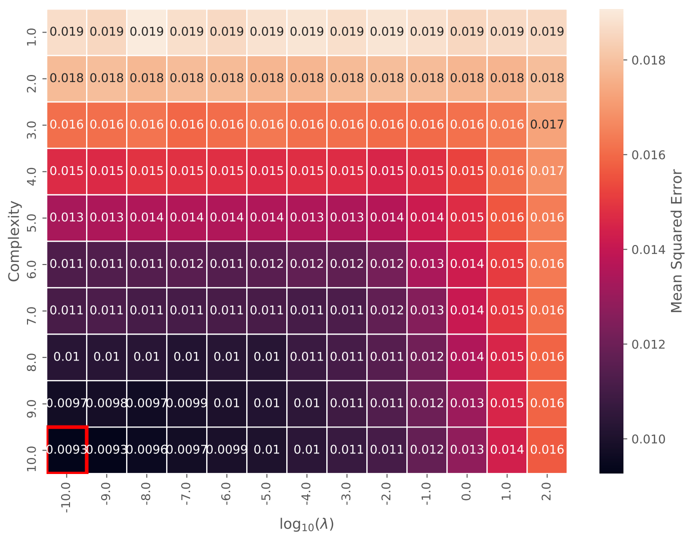

Figure 7.1. Heat map of the estimated MSE of Ridge method on the terrain data for value ranges of λ and model complexity d.

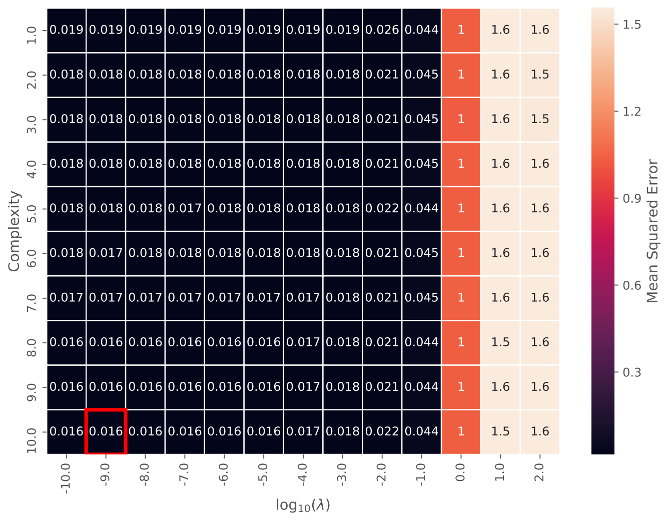

Figure 7.2. Heat map of the estimated MSE of the Lasso method on the terrain data for value ranges of λ and model complexity d.

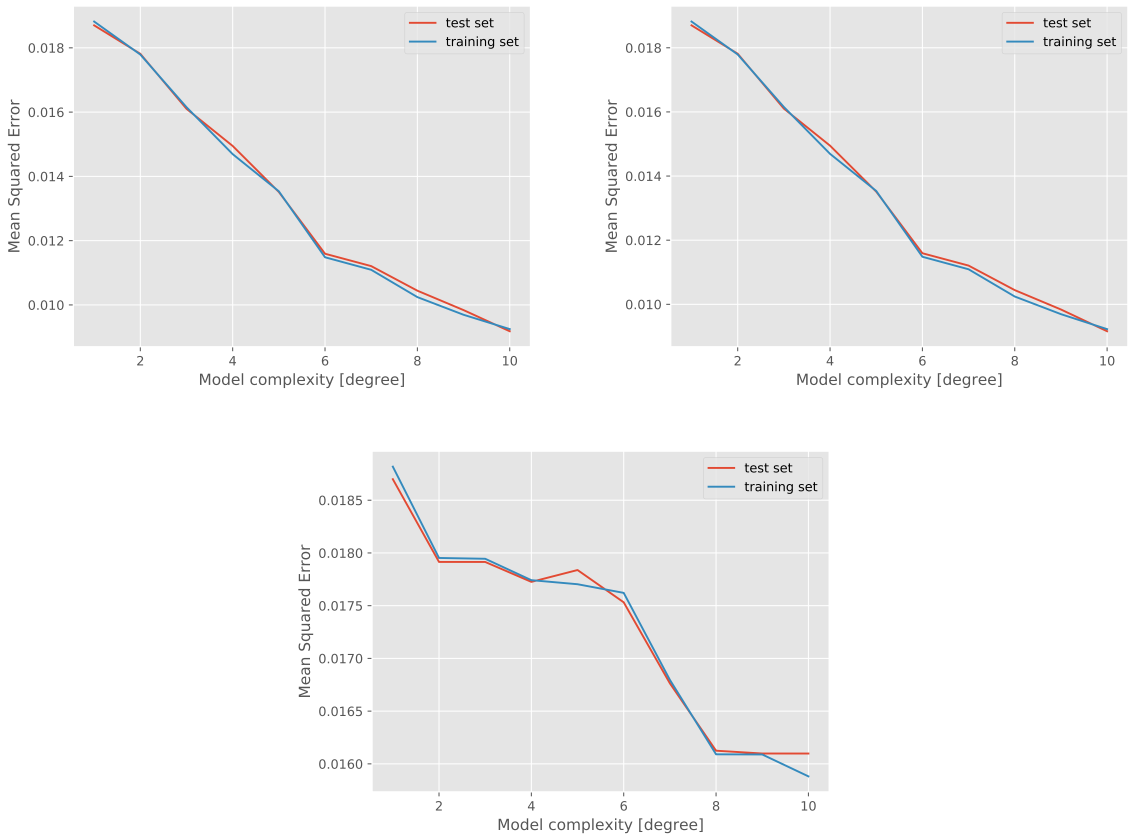

Figure 7.3. MSE of the predictions by the OLS(top left), Ridge(top right) and Lasso(bottom) methods on the training and testing portions of the terrain data for different model complexities. $\lambda_{Ridge} = 10^{-10}$, $\lambda_{Lasso} = 10^{-9}$.